Neural Networks Based on Competition

In some examples of pattern classification we encountered a situation in which the net was trained to classify the input signal into one of the output categories, while the net responded to more than one.

In circumstances such as this, in which only one of several neurons should respond, we can include additional structure in the network so that the net is forced to make a decision as to which one unit will respond. The mechanism by which this is achieved is called ***competition***.

The most extreme form of competition among a group of neurons is called ***Winner Take All***. As the name suggests, only one neuron in the competing group will have a **nonzero output signal** when the competition is completed. Many neural nets use the idea of competition among neurons to enhance the contrast in activations of the neurons.

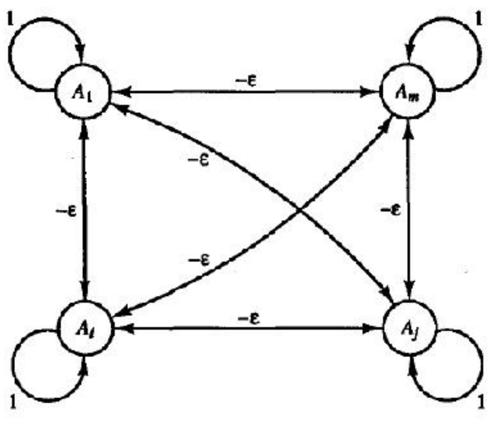
1- FIXED-WEIGHT COMPETITIVE NETS

1.1 MAXNET

MAXNET [Lippmann, 1987] is a specific example of a neural net based on competition. It can be used as a subnet to pick the node whose input is the largest. The *m* nodes in this subnet are completely interconnected, with symmetric weights.

There is no training algorithm for the MAXNET; the weights are fixed.

The architecture of MAXNET is as shown in the figure (1):



The activation function for the MAXNET is

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

The application procedure is as follows:

1- Initialize activations and weights ($0 < \varepsilon < 1/m$)

$a_j(0)$ input to node $\mathbf{A_j}$

$$w_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\varepsilon & \text{if } i \neq j \end{cases}$$

2- Update the activation of each node: $j = 1, \dots, m$

$$a_j(new) = f\,[a_j(old) - \varepsilon \sum_{k \neq j} a_k(old)]$$

3- Save activation for use in next iteration

$$a_j(old) = a_j(new), \quad j = 1, \dots, m$$

4- If more than one node has a nonzero activation, then updating the activation will continue, otherwise we should stop.

Note that in Step 2, the input to the function f is simply the total input to node $\mathbf{A}_j$ from all nodes, including itself.

Example -1: Consider the action of a MAXNET with four neurons and inhibitory weights $\varepsilon = 0.2$. The neurons given initial activation (input signal)

$a_1(0) = 0.2$, $a_2(0) = 0.4$, $a_3(0) = 0.6$, $a_4(0) = 0.8$

sol:

$$a_j(new) = f\,[a_j(old) - \varepsilon \sum a_k(old)]$$

$$a_1(new) = f\,[a_1(old) - 0.2(a_2(old) + a_3(old) + a_4(old)]$$

$$a_2(new) = f\,[a_2(old) - 0.2(a_1(old) + a_3(old) + a_4(old)]$$

$$a_3(new) = f\,[a_3(old) - 0.2(a_2(old) + a_2(old) + a_4(old)]$$

$$a_4(new) = f\,[a_4(old) - 0.2(a_1(old) + a_2(old) + a_3(old)]$$

$$a_1(1) = f\,[a_1(0) - 0.2(a_2(0) + a_3(0) + a_4(0)]$$

$$= f\,[0.2 - 0.2(0.4 + 0.6 + 0.8)] = f\,[-\,0.16] = 0$$

$$a_2(1) = f\,[a_2(0) - 0.2(a_1(0) + a_3(0) + a_4(0)]$$

$$= f\,[0.4 - 0.2(0.2 + 0.6 + 0.8)] = f\,[0.08] = 0.08$$

$$a_3(1) = f\,[a_3(0) - 0.2(a_1(0) + a_2(0) + a_4(0)]$$

$$= f\,[0.6 - 0.2(0.2 + 0.4 + 0.8)] = f\,[0.32] = 0.32$$

$$a_4(1) = f\,[a_4(0) - 0.2(a_1(0) + a_2(0) + a_3(0)]$$

$$= f\,[0.8 - 0.2(0.2 + 0.4 + 0.6)] = f\,[0.56] = 0.56$$

The activations found as the net iterates are

$a_l(l) = 0.0$	$a_2(l) = 0.08$	$a_3(l) = 0.32$	$a_4(l) = 0.56$
$a_l(2) = 0.0$	$a_2(2) = 0.0$	$a_3(2) = 0.192$	$a_4(2) = 0.48$
$a_l(3) = 0.0$	$a_2(3) = 0.0$	$a_3(3) = 0.096$	$a_4(3) = 0.442$
$a_l(4) = 0.0$	$a_2(4) = 0.0$	$a_3(4) = 0.008$	$a_4(4) = 0.422$
$a_1(5) = 0.0$	$a_2(5) = 0.0$	$a_3(5) = 0.0$	$a_4(5) = 0.421$

1.2 Mexican Hat

The Mexican Hat network [Kohonen, 1989a] is a more general contrast-enhancing subnet than the MAXNET. Three types of links can be found in such network:

1- Each neuron is connected with excitatory (positively weighted) links to a number of "cooperative neighbors," neurons that are in close proximity.

2- Each neuron is also connected with inhibitory links (with negative weights) to a number of "competitive neighbors," neurons that are somewhat further away.

3- There may also be a number of neurons, further away still, to which the neuron is not connected.

All of these connections are within a particular layer of a neural net, so, as in the case of MAXNET, the neurons receive an external signal in addition to these interconnection signals.

The pattern of interconnections just described is **repeated for each neuron** in the layer. The interconnection pattern for unit X_i is illustrated in Figure 2. For ease of description, the neurons are pictured as arranged in a linear order, with:

- **Positive connections** *between unit X_i and neighboring units one or two positions on either side*;
- **Negative connections** *are shown for units three positions on either side*.
- The size of the region of cooperation (positive connections) and the region of competition (negative connections) may vary.
- An external signal $\mathbf{s_i}$ is applied to unit X_i. The contrast enhancement of the signals received by unit X_i is accomplished by iteration for several time steps. The activation of unit X_i at time t is given by:

$$x_i(t) = f\,[s_i(t) + \sum_k w_k x_{i+k}(t - 1)],$$

where the terms in the summation are the weighted signals from other units (cooperative and competitive neighbors) at the previous time step.

In the example illustrated in Figure 2, the weight w_k from unit X_i to unit X_{i+k} is **positive for k = -2, -1, 0, 1, and 2**, **negative for k =-3, and 3**, and **zero for units beyond these**.

Architecture:

- The interconnections for the Mexican Hat net involve two symmetric regions around each individual neuron.
- The connection weights within the closer region weights between a typical unit X_i and units X_{i+1}, X_{i+2}, X_{i-l} , and X_{i-2}, for example, are positive (and often are taken to have the same value). These weights are shown as w_1 and w_2 in figure 2. The weights between X_i and units X_{i+3} and X_{i-3} are negative (shown as $\mathbf{w_3}$ in the figure).

And Unit X_i is not connected to units $\boldsymbol{X_{i-4}}$ and $\boldsymbol{X_{i+4}}$ in this sample architecture.

In other word, In the illustration, units within a radius of 2 to the typical unit X_i are connected with positive weights; units within a radius of 3, but outside the radius of positive connections, are connected with negative weights; and units further than 3 units away are not connected.

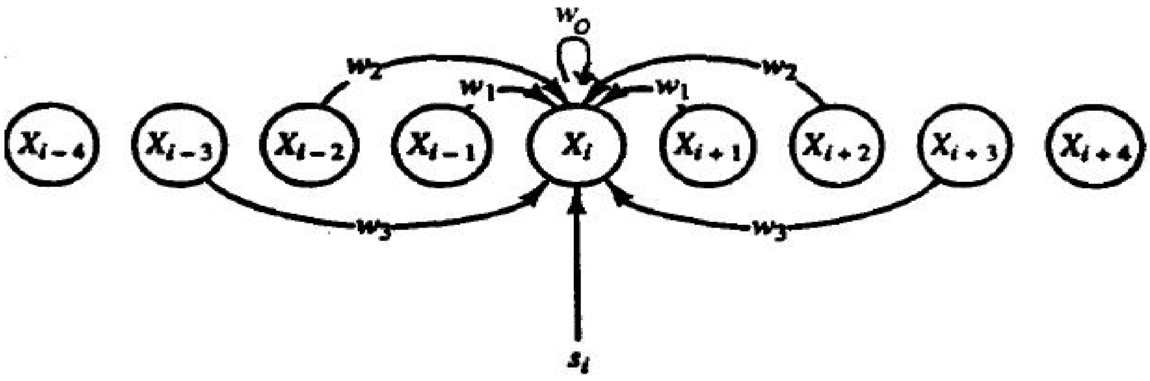


Figure 2 Mexican Hat interconnections for unit *X*,.

Algorithm

The nomenclature we use is as follows:

R2 Radius of region of interconnections; $\boldsymbol{X_i}$ is connected to units X_{i+k} and $\boldsymbol{X_{i-k}}$ for k = 1, ... ,*R2*·

R1 Radius of region with positive reinforcement; *R1* < *R2* .

$\mathbf{w_k}$ Weight on interconnections between X_i and units X_{i+k} and X_{i-k} :

w_k is positive for $0 \le k \le R1$

w_k is negative for $R1 < k \le R2$

x Vector of activations.

x_oId Vector of activations at previous time step.

t_max Total number of iterations of contrast enhancement.

s External signal.

As presented, the algorithm corresponds to the external signal being given only for the first iteration (Step 2) of the contrast-enhancing iterations. We have:

1- Initialize parameters *t_max, R1, R2* as desired.

Initialize weights:

w_k = C_1 for $k = 0, \dots, R1$, ($C_1 > 0$)

w_k = C2 for $k = R1 + 1, \dots, R2$ ($C2 < 0$).

Initialize x-old to 0.

2- Present external signal s:

x = s.

Save activations in array x-old (for $i = 1, \dots, n$):

$x\_old_i = x_i$.

Set iteration counter: $t = 1$.

3- While *t* is less than t_max, do Steps 4-8.

4- Compute net input ($i = 1, \dots, n$):

$$xi = C_1 \sum_{k=-R1}^{R1} x\_old_{i+k} + C_2 \sum_{k=-R2}^{-R-1} x\_old_{i+k} + C_2 \sum_{k=R1+1}^{R2} x\_old_{i+k}$$

5- Apply activation function (ramp function from 0 to *x_max*, slope 1):

$$\mathbf{x_i = \min(x\_max, \max(0, x_i));\quad (i = 1, \dots, n).}$$

6- Save current activations in x_old:

$$\mathbf{x\_old_i = x_i\ ;\ (i = 1, \dots, n).}$$

7- increment iteration counter

$$t = t + 1$$

8- Test stopping condition

If $t < t\_max$, continue; otherwise, stop.

The positive reinforcement from nearby units and negative reinforcement from units that are further away have the effect of increasing the activation of units with larger initial activations and reducing the activations of those that had a smaller external signal. This is illustrated in the following example.

Example-2: Using the Mexican Hat Algorithm We illustrate the Mexican Hat algorithm for a simple net with seven units. The activation function for this net is:

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 2 \\ 2 & \text{if } x > 2 \end{cases}$$

R1 = 1, R2 = 2

C1 = 0.6, C2 = - 0.4

The external signal s is (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0)

Sol:

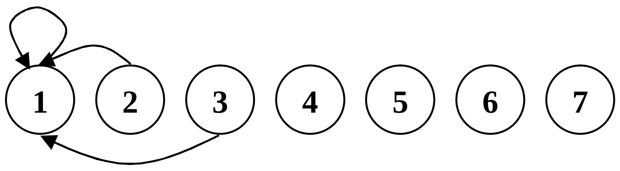
$x_i = s_i$

x = (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0).

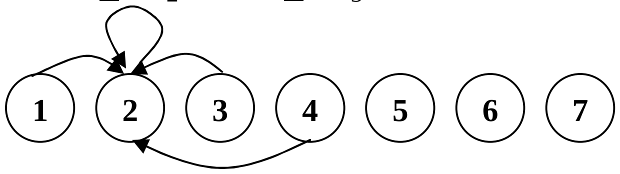
Save in x_old:

x_old = (0.0, 0.5, 0.8, 1.0, 0.8, 0.5, 0.0).

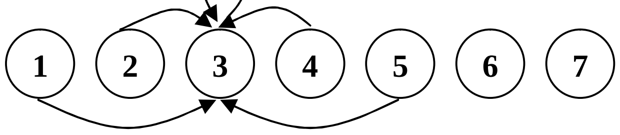
$$xi = C_1 \sum_{k=-R1}^{R1} x\_old_{i+k} + C_2 \sum_{k=-R2}^{-R-1} x\_old_{i+k} + C_2 \sum_{k=R1+1}^{R2} x\_old_{i+k}$$



$x_1 = 0.6\ x\_old_1 + 0.6\ x\_old_2 - 0.4\ x\_old_3$



$x_2 = 0.6\ x\_old_1 + 0.6\ x\_old_2 + 0.6\ x\_old_3 - 0.4\ x\_old_4$



$x_3 = -0.4x\_old_1 + 0.6x\_old_2 + 0.6x\_old_3 + 0.6x\_old_4 - 0.4x\_old_5$

$x_4 = -0.4x\_old_2 + 0.6x\_old_3 + 0.6x\_old_4 + 0.6x\_old_5 - 0.4x\_old_6$

$x_5 = -0.4x\_old_3 + 0.6x\_old_4 + 0.6x\_old_5 + 0.6x\_old_6 - 0.4x\_old_7$

x_6 = - 0.4 x_old$_4$ + 0.6 x_old$_5$ + 0.6 x_old$_6$ + 0.6 x_old$_7$

x_7 = -0.4 x_old$_5$ + 0.6 x_old$_6$ + 0.6 x_old$_7$

***t* = 1**

x_l = 0.6(0.0) + 0.6(0.5) - 0.4(0.8) = -0.2

x_2 = 0.6(0.0) + 0.6(0.5) + 0.6(0.8) - 0.4(1.0) = 0.38

x_3 = -0.4(0.0) + 0.6(0.5) + 0.6(0.8) + 0.6(1.0) - 0.4(0.8) = 1.06

x_4 = -0.4(0.5) + 0.6(0.8) +,0.6(1.0) + 0.6(0.8) - 0.4(0.5) = 1.16

x_5 = -0.4(0.8) + 0.6(1.0) + 0.6(0.8) + 0.6(0.5) - 0.4(0.0) = 1.06

x_6 = -0.4(1.0) + 0.6(0.8) + 0.6(0.5) + 0.6(0.0) = 0.38

x_7 = -0.4(0.8) + 0.6(0.5) + 0.6(0.0) = -0.2.

x = (0.0, 0.38, 1.06, 1.16, 1.06, 0.38, 0.0).

***t* = 2**

x_l = 0.6(0.0) + 0.6(0.38) - 0.4(1.06) = -0.196

x_2 = 0.6(0.0) + 0.6(0.38) + 0.6(1.06) - 0.4(1.16) = 0.39

x_3 = -0.4(0.0) + 0.6(0.38) + 0.6(1.06) + 0.6(1.16) - 0.4(1.06)= 1.14

x_4 = -0.4(0.38) + 0.6(1.06) + 0.6(1.16) + 0.6(1.06) - 0.4(0.38) = 1.66

x_5 = -0.4(1.06) + 0.6(1.16) + 0.6(1.06) + 0.6(0.38) - 0.4(0.0) = 1.14

x_6 = -0.4(1.16) + 0.6(1.06) + 0.6(0.38) + 0.6(0.0) = 0.39

x_7 = -0.4(1.06) + 0.6(0.38) + 0.6(0.0) = -0.196

x = (0.0, 0.39, 1.14, 1.66, 1.14, 0.39, 0.0).

The pattern of activations is shown for *t* = 0, 1, and *2* in Figure 3.

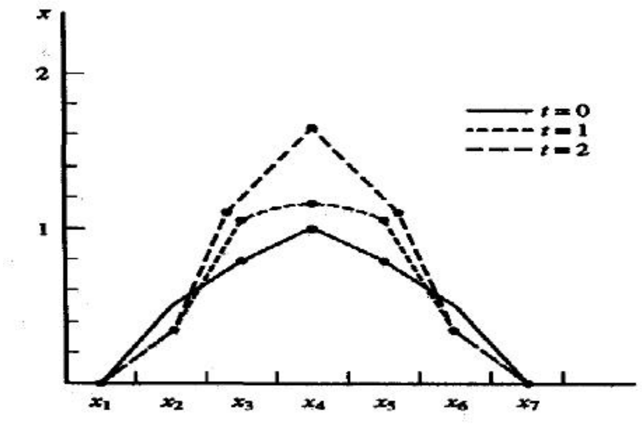


Figure (3) Result for Mexican Hat

1.3 Hamming Net

A Hamming net [Lippmann, 1987; DARPA, 1988] is a maximum likelihood classifier net that can be used to determine which of several exemplar vectors is most similar to an input vector (an n-tuple).

- The exemplar vectors determine the weights of the net.
- The measure of similarity between the input vector and the stored exemplar vectors is *n* minus the Hamming distance between the vectors.
- The Hamming distance between two vectors ***d*** is the number of components in which the vectors differ.

However, if ***n*** is the number of components in the vectors, and ***a*** is the number of components in which the vectors agree then,

$$d = n - a$$

For bipolar vectors **x** and **y**,

$$\mathbf{x}.\mathbf{y}^T = a - d,$$

$$\mathbf{x}.\mathbf{y}^T = 2a - n$$

or

$$2a = \mathbf{x}.\mathbf{y} + n$$

By setting the ***weights*** to be ***one-half the exemplar vector*** and setting the value of the ***bias to n/2***, the net will find the unit with the ***closest exemplar*** simply by finding the unit with the ***largest net input***.

The Hamming net uses **MAXNET** as a subnet to find the unit with the ***largest net input***.

The lower net consists of ***n*** input nodes, each connected to ***m*** output nodes (where ***m*** is the number of exemplar vectors stored in the net).

The output nodes of the lower net feed into an upper net (MAXNET) that calculates the best exemplar match to the input vector.

The input and exemplar vectors are bipolar.

Architecture

The sample architecture shown in Figure 4 assumes input vectors are 4-tuples, to be categorized as belonging to one of two classes.

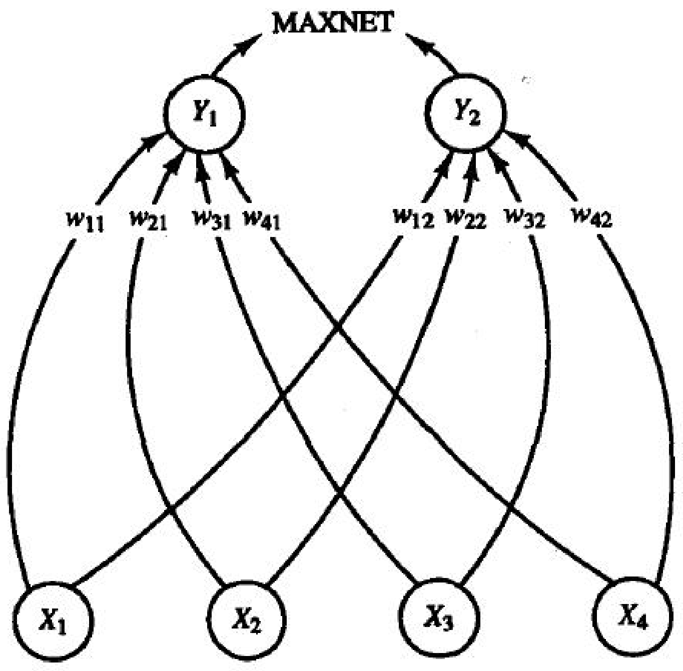

MAXNET
Y_1
Y_2
w_{11} w_{21} w_{31} w_{41}
w_{12} w_{22} w_{32} w_{42}
X_1
X_2
X_3
X_4

Figure (4) The Hamming net

Application

Given a set of ***m*** bipolar exemplar vectors, ***e(l), e(2), ... , e(m)***,

The Hamming net can be used to find the exemplar that is closest to the bipolar input vector **x**.

The net input ***y_in*** to unit ***Y_j;***, gives the ***agree*** number of components between the input vector and the exemplar vector for unit **Y_j e(j)**, ***(a = n– d),*** where,

d is the Hamming distance between these vectors;

n number of input nodes, number of components of any input vector;

m number of output nodes, number of exemplar vectors;

e(j) the jth exemplar vector: j =1, 2, … , m;

e(j) = (e1(j), ... , ei(j), ... , en(j))

The application procedure for the Hamming net is:

1- To store the ***m*** exemplar vectors, initialize the weights:

$$w_{ij} = e_i(j)/2 \;,\; (i = 1, \ldots, n; j = 1, \ldots, m).$$

And initialize the biases:

$$b_j = n/2, \; (j = 1, \ldots, m).$$

2- For each vector x, Compute the net input to each unit Y_j

$$y\_in_j = b_j + \sum_i x_i w_{ij}, \qquad (j=1, \ldots, m)$$

3- Initialize activations for MAXNET:

$$y_j(0) = y\_in_j, \qquad (j=1, \ldots, m)$$

4- MAXNET iterates to find the best match exemplar.

Example-3: A Hamming net to cluster four vectors

Given the exemplar vectors:

e(1) = (1, -1, -1, -1,), e(2) = (-1, -1, -1, 1)

Using the Hamming net to find the exemplar that is closest to each of the bipolar input pattern:

(1, 1, -1, -1), (1, -1, -1, -1), (-1, -1, -1, 1), and (-1, -1, 1, 1)

Sol: Store the m exemplar vectors in the weights:

$$w_{ij} = e_i(j)/2 \;,\; (i = 1, \ldots, n; j = 1, \ldots, m$$

$$\mathbf{W} = \begin{bmatrix} .5 & -.5 \\ -.5 & -.5 \\ -.5 & -.5 \\ -.5 & .5 \end{bmatrix}$$

Initialize the biases

$$b_j = n/2, \; (j = 1, \ldots, m).$$

$$b_1 = b_2 = 4/2 = 2$$

For the first input vector x = (1, 1, -1, -1)

$$y\_in_1 = b_1 + \sum_i x_i w_{i1}$$

$$= 2 + (1*.5 + 1*(-.5) + (-1)*(-.5) + (-1)*(-.5) = 2 + 1 = 3$$

$$y\_in_2 = b_2 + \sum_i x_i w_{i2}$$

= 2 + (1*(-.5) +1 *(-.5) + (-1)*(-.5) + (-1)*(.5) = 2 - 1 = 1

These values represent the Hamming similarity because **(1,1,-1,-1) agrees** with **e(1)=(l,-1,-1,-1)** in the first, third, and fourth components and because **(1, 1 , - 1, - 1) agrees** with **e(2)** = (**- 1, - 1, - 1, 1)** in only the third component.

$y_1(0) = 3$, $y_2(0) = 1$

* Since ***$y_1(0) > y_2(0)$***, **MAXNET** will find that unit Y_1 has the best match exemplar for input vector **x** = (1, 1, - 1, - 1).

For the 2nd input vector x = (l,-1,-1,-l),

y_in_1 = 2 + (1*.5 +(-1)*(-.5) + (-1)*(-.5) + (-1)*(-.5) = 2 + 2 = 4

y_in_2 = 2 + (1*(-.5) +(-1)*(-.5) + (-1)*(-.5) + (-1)*(.5) = 2 + 0 = 2

Note that the input vector **agrees** with **e(1)** in all four components and agrees with **e(2)** in the second and third components.

$y_1(0) = 4$, $y_2(0) = 2$

* Since ***$y_1(0) > y_2(0)$***, **MAXNET** will find that unit Y_1 has the best match exemplar for input vector **x** = (1, -1, - 1, - 1).

For the 3rd. input vector x = (-l,-1,-1,l),

y_in_1 = 2 + ((-1)*.5 +(-1)*(-.5) + (-1)*(-.5) + 1*(-.5) = 2 + 0 = 2

y_in_2 = 2 + ((-1)*(-.5) +(-1)*(-.5) + (-1)*(-.5) + 1*(.5) = 2 + 2 = 4

The input vector **agrees** with **e(1)** in the 2nd. And 3rd. components and agrees with **e(2)** in all four components.

$y_1(0) = 2$, $y_2(0) = 4$

* Since ***$y_2(0) > y_1(0)$***, **MAXNET** will find that unit Y_2 has the best match exemplar for input vector **x** = (-1, -1, - 1, 1).

For the 4th. input vector x = (-l,-1,1,l),

y_in_1 = 2 + ((-1)*.5 +(-1)*(-.5) + 1*(-.5) + 1*(-.5) = 2 - 1 = 1

y_in_2 = 2 + ((-1)*(-.5) +(-1)*(-.5) + 1*(-.5) + 1*(.5) = 2 + 1 = 3

The input vector **agrees** with **e(1)** in the 2nd. component and agrees with **e(2)** in the 1st, 2nd., and 4th. components.

$\boldsymbol{y_1(0) = 1,} \qquad \boldsymbol{y_2(0) = 3}$

* Since $\boldsymbol{y_2(0) > y_1(0)}$, **MAXNET** will find that unit Y_2 has the best match exemplar for input vector **x** = (-1, -1, 1, 1).